IES de Teis
Avda. de Galicia, 101
36216 – Vigo

Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es

WordCount en Python usando MRJob

Python con MRJob

MRJob es una librería Python ampliamente utilizada para la implementación de “mappers” y “reducers”, cuenta con una buena comunidad de usuarios y está bien documentada. Las ventajas que aporta es que mantiene el “mapper” y el “reducer” en una sola clase.

Otra de las diferencias es que tanto el código de “mapper” como el código del “reducer” están en el mismo archivo. Por defecto se ejecutará “mapper” y “reducer” en este orden pero hay ocasiones en las que nos interesa ejecutar dos “mappers” distintos seguidos y después un “reducer”, o tal vez queramos enlazar dos trabajos MapReduce consecutivos. Para cualquier escenario de este tipo simplemente tendremos que indicar la secuencia de fases en un método llamado “step”

Otra funcionalidad interesante es que podemos probar el código en local sin necesidad de un clúster Hadoop, para ello simplemente tendremos que lanzar el archivo con python3 o darle permisos de ejecución.

Instalación

Necesitaremos “pip”, el gestor de paquetes de Python, para instalar la librería “mrjob”. Si aun no tenemos en nuestro sistema el gestor de paquetes Python “pip” debemos instalarlo mediante el siguiente comando:

```
sudo apt install pip
```

Nos aseguramos de que tenemos pip instalado consultando su versión con

```
pip --version
```

La salida debería ser la versión de pip instalada. Ahora sí ya podemos instalar MRJob con el siguiente comando:

```
pip install mrjob
```

```
administrador@nodo01:~/trabajo01$ pip install mrjob
Defaulting to user installation because normal site-packages is not writeable
Collecting mrjob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
     |████████████████████████████████| 439.6/439.6 KB 5.0 MB/s eta 0:00:00
Requirement already satisfied: PyYAML>=3.10 in /usr/lib/python3/dist-packages (from mrjob) (5.4.1)
Installing collected packages: mrjob
  WARNING: The scripts mrjob, mrjob-3 and mrjob-3.7 are installed in '/home/administrador/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed mrjob-0.7.4
administrador@nodo01:~/trabajo01$
```

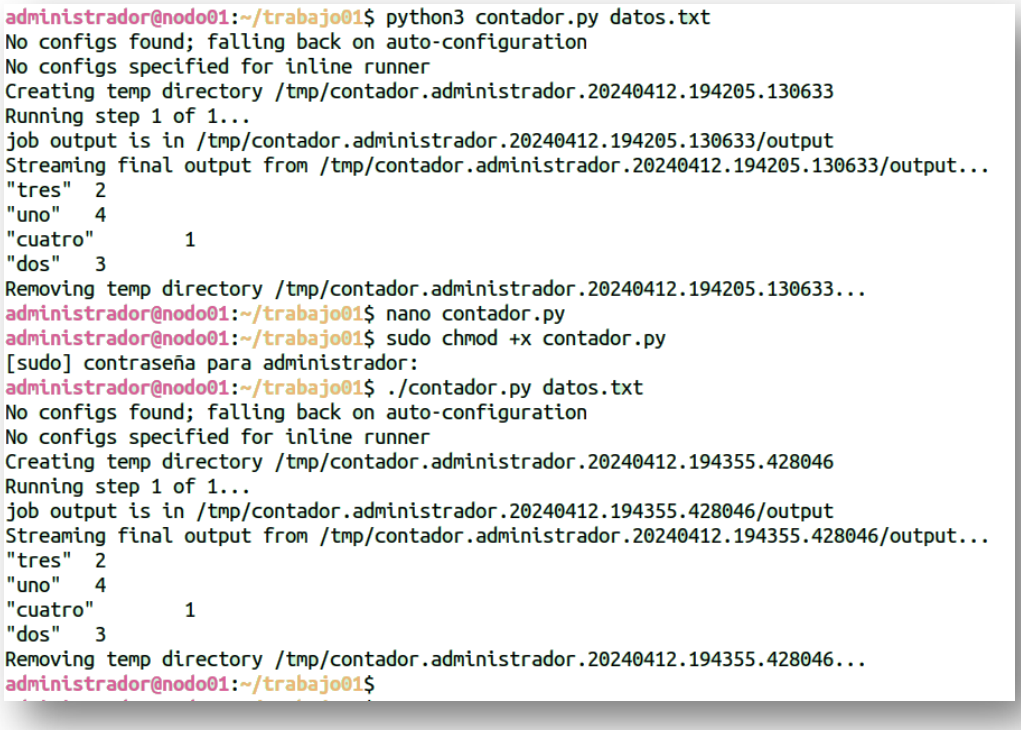
Implementación del map y reduce

Una vez instalado ya podemos implementar nuestro “mapper” y “reducer”. Podemos hacerlo desde cualquier editor de texto. La idea es heredar de la clase MRJob y sobrescribir sus métodos “mapper” y “reducer”. En el “mapper” tendremos que pasarle la referencia a “self”, un campo vacío (se usa si hubiera una sucesión de “mappers” distintos) y line que es por dónde irán llegando los datos.

```
from mrjob.job import MRJob

class MiContador(MRJob):

   def mapper(self,_,line):
      for word in line.split()
         yield (word,1(

   def reducer(self, key, values):
     yield (key, sum(values))

if __name__==”__main__”:
   MiContador.run()
```

Como podemos ver en el código, una de las diferencias más importantes es que aquí usamos el comando “yield” y no “print” para pasar los pares clave-valor a otras etapas.

Ejecución en local



```
administrador@nodo01:~/trabajo01$ python3 contador.py datos.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/contador.administrador.20240412.194205.130633
Running step 1 of 1...
job output is in /tmp/contador.administrador.20240412.194205.130633/output
Streaming final output from /tmp/contador.administrador.20240412.194205.130633/output...
"tres"  2
"uno"   4
"cuatro"        1
"dos"   3
Removing temp directory /tmp/contador.administrador.20240412.194205.130633...
administrador@nodo01:~/trabajo01$ nano contador.py
administrador@nodo01:~/trabajo01$ sudo chmod +x contador.py
[sudo] contraseña para administrador:
administrador@nodo01:~/trabajo01$ ./contador.py datos.txt
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory /tmp/contador.administrador.20240412.194355.428046
Running step 1 of 1...
job output is in /tmp/contador.administrador.20240412.194355.428046/output
Streaming final output from /tmp/contador.administrador.20240412.194355.428046/output...
"tres"  2
"uno"   4
"cuatro"        1
"dos"   3
Removing temp directory /tmp/contador.administrador.20240412.194355.428046...
administrador@nodo01:~/trabajo01$
```

```
python3 contador.py datos.txt
```

Ojo con la salida porque es efímera. La carpeta local donde se ha guardado las salidas de los “reducers” se borra nada más mostrar los datos por pantalla.

Ejecución en Hadoop

En caso de querer ejecutar el trabajo en un entorno Hadoop tenemos que indicarlo mediante el parámetro “-r hadoop”

```
Python3 contador.py-r hadoop hdfs:///datos.txt \
--jobconf mapreduce.job.priority=VERY_HIGH \
-- jobconf mapreduce.job.maps=2 \
-- jobconf mapreduce.job.reduces=1 \
```

Con la primera línea es suficiente ya que tenemos el código del “mapper” y “reducer”, la indicación de que use hadoop y la ruta HDFS de los datos de entrada. El resto de las líneas son opcionales pero pueden resultar interesantes para comprobar el rendimiento de Hadoop y unos datos de prueba modificando el número de “mappers” y “reducers”.

IES de Teis
Avda. de Galicia, 101
36216 – Vigo

Tfno: 886 12 04 64
e-mail: ies.teis@edu.xunta.es
http://www.iesteis.es


```
                Data-local map tasks=2
                Launched map tasks=2
                Launched reduce tasks=1
                Total megabyte-milliseconds taken by all map tasks=6714368
                Total megabyte-milliseconds taken by all reduce tasks=1931264
                Total time spent by all map tasks (ms)=6557
                Total time spent by all maps in occupied slots (ms)=6557
                Total time spent by all reduce tasks (ms)=1886
                Total time spent by all reduces in occupied slots (ms)=1886
                Total vcore-milliseconds taken by all map tasks=6557
                Total vcore-milliseconds taken by all reduce tasks=1886
        Map-Reduce Framework
                CPU time spent (ms)=1100
                Combine input records=0
                Combine output records=0
                Failed Shuffles=0
                GC time elapsed (ms)=193
                Input split bytes=160
                Map input records=4
                Map output bytes=85
                Map output materialized bytes=117
                Map output records=10
                Merged Map outputs=2
                Peak Map Physical memory (bytes)=313516032
                Peak Map Virtual memory (bytes)=2552492032
                Peak Reduce Physical memory (bytes)=227135488
                Peak Reduce Virtual memory (bytes)=2553470976
                Physical memory (bytes) snapshot=851558400
                Reduce input groups=4
                Reduce input records=10
                Reduce output records=4
                Reduce shuffle bytes=117
                Shuffled Maps =2
                Spilled Records=20
                Total committed heap usage (bytes)=665321472
                Virtual memory (bytes) snapshot=7657906176
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
job output is in hdfs:///user/administrador/tmp/mrjob/contador.administrador.20240417.174207.228882/output
Streaming final output from hdfs:///user/administrador/tmp/mrjob/contador.administrador.20240417.174207.228882/output...
"cuatro"        1
"dos"   3
"tres"  2
"uno"   4
Removing HDFS temp directory hdfs:///user/administrador/tmp/mrjob/contador.administrador.20240417.174207.228882...
Removing temp directory /tmp/contador.administrador.20240417.174207.228882...
administrador@nodo01:~/trabajo01$
```

Si nos fijamos bien en la salida podemos ver como el resultado es efímero. Es decir, se crea una carpeta para guardar el resultado de los “mappers”, se muestra por pantalla el resultado y por último se borran los datos

```
python3 contador.py -r hadoop hdfs:///datos.txt > solución.txt
```

En este comando anterior te muestro la manera más sencilla de guardar los datos en local, haciendo una redirección de la salida a un archivo local.

Si queremos mantener el resultado en HDFS de manera permanente debemos indicarlo con el parámetro “-o” y la ruta HDFS de la carpeta donde se guardarán permanentemente los datos.

Problemas en un clúster Hadoop

El mayor problema con el que nos enfrentamos al hacer aplicaciones que usen librerías es como hacer que los nodos las reconozcan. La opción sencilla es instalarlas en todos los nodos y hacer, de esa manera, que estén accesibles para todos los contenedores de esos nodos.